SVEUČILIŠTE U SPLITU  
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

IZRADA WEB APLIKACIJE ZA PRAĆENJE TIJEKA PROJEKTA

Lorena Šegvić

Split, rujan 2018.

SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Diplomski studij:  **Računarstvo**

Smjer/Usmjerenje:

Oznaka programa: **250**

Akademska godina: **2017/2018.**

Ime i prezime: **LORENA ŠEGVIĆ**

Broj indeksa: **819-2015**

**ZADATAK DIPLOMSKOG RADA**

Naslov: **Izrada web aplikacije za praćenje tijeka projekta**

Zadatak: U okviru ovog diplomskog rada potrebno je napraviti web aplikaciju koja će omogućiti operateru unos evidencije za projekte na kojima radi (partner, program, projekt, izvršitelj, vrsta posla, količina, j. mjere, opis). U aplikaciji treba postojati više vrsta korisnika s različitim ovlastima (operater, voditelj, administrator). Opisati korištene tehnologije, arhitekturu aplikacije i postupak njene izrade.

Prijava rada: 20.11.2017.

Rok za predaju rada: 18.09.2018.

Rad predan: 13.09.2018.

Predsjednik

Odbora za diplomski rad: Mentor:

Prof. dr. sc. Sven Gotovac Izv. prof. dr. sc. Stipo Čelar

**IZJAVA**

Ovom izjavom potvrđujem da sam završni rad s naslovom „**IZRADA WEB APLIKACIJE ZA PRAĆENJE TIJEKA PROJEKTA**“ pod mentorstvom izv. prof. dr. sc. Stipe Čelara pisala samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući navela u diplomskom radu citirala sam i povezala s korištenim bibliografskim jedinicama.

Studentica

Lorena Šegvić

**SADRŽAJ**

[1. UVOD 2](#_Toc524462247)

[2. SLIČNI ALATI ZA PRAĆENJE TIJEKA PROJEKTA 3](#_Toc524462248)

[2.1 Microsoft Project 3](#_Toc524462249)

[2.2 Trello 4](#_Toc524462250)

[2.3 Kanban Flow 5](#_Toc524462251)

[3. METODE I TEHNOLOGIJE KORIŠTENE U IZRADI APLIKACIJE 8](#_Toc524462252)

[3.1 ASP.NET 8](#_Toc524462253)

[3.2 Obrazac oblikovanja model-view-controller 9](#_Toc524462254)

[3.2.1 Upravitelj 10](#_Toc524462255)

[3.2.2 Model 11](#_Toc524462256)

[3.2.3 Prikaz 11](#_Toc524462257)

[3.2.4 Prednosti i nedostaci 12](#_Toc524462258)

[3.3 Entity framework 12](#_Toc524462259)

[3.4 LINQ upiti 13](#_Toc524462260)

[3.5 Database-first, Code-first, Model-first 14](#_Toc524462261)

[3.6 Struktura direktorija aplikacije 15](#_Toc524462262)

[3.7 Izrada korisničkog sučelja 18](#_Toc524462263)

[3.8 ASP.NET URL rutiranje 19](#_Toc524462264)

[4. KREIRANJE MVC WEB APLIKACIJE U .NET OKRUŽENJU 21](#_Toc524462265)

[4.1 Zahtjevi na aplikaciju 21](#_Toc524462266)

[4.2 ASP .NET Identity 22](#_Toc524462267)

[4.3 Vrste korisničkih uloga 22](#_Toc524462268)

[4.4 Validacija podataka 25](#_Toc524462269)

[4.5 Popunjavanje baze podataka korištenjem Code-First pristupa 28](#_Toc524462270)

[4.6 Dodavanje modela 29](#_Toc524462271)

[4.7 View Modeli 31](#_Toc524462272)

[4.8 Stvaranje baze podataka 32](#_Toc524462273)

[4.9 Scaffolding 33](#_Toc524462274)

[4.10 Sinkronizacija modela s bazom podataka 34](#_Toc524462275)

[4.11 Način rada upravitelja 35](#_Toc524462276)

[4.12 Prenošenje podataka iz upravitelja u prikaz 36](#_Toc524462277)

[4.13 Izgled aplikacije 37](#_Toc524462278)

[5. ZAKLJUČAK 41](#_Toc524462279)

[6. LITERATURA 42](#_Toc524462280)

[7. SAŽETAK/ABSTRACT I KLJUČNE RIJEČI/KEYWORDS 45](#_Toc524462281)

[8. PRILOZI 47](#_Toc524462282)

[8.1 Kazalo slika, tablica i kodova 47](#_Toc524462283)

[8.1.1 Kazalo slika 47](#_Toc524462284)

[8.1.2 Kazalo tablica 47](#_Toc524462285)

[8.1.3 Kazalo kodova 48](#_Toc524462286)

[8.2 Popis oznaka i kratica 48](#_Toc524462287)

# UVOD

Vođenje projekata može biti veoma stresan posao, pogotovo ako programeri i dizajneri drastično premašuju zadane rokove. U današnje vrijeme razni softverski alati postaju nezaobilazni u projektnom menadžmentu, a svaki od njih je specifičan, kao i svaki projekt. U toj činjenici leži i motivacija za odabranu temu. Cilj rada je napraviti alat koji će pratiti razvoj projekata i olakšati cijeli proces nadziranja svake faze projekta. Jedan od alata koje voditelji projekata danas često koriste je tzv. dijagram toka (engl. *flow chart*) koji im pruža jasnu sliku projekta te grafički prikazuje ciljeve i nastoji logički rasporediti aktivnosti unutar njega. Isto tako pomaže u praćenju razvoja projekata i eventualnim statusnim izvještajima. To je ono što ćemo pokušati postići izradom ove aplikacije.

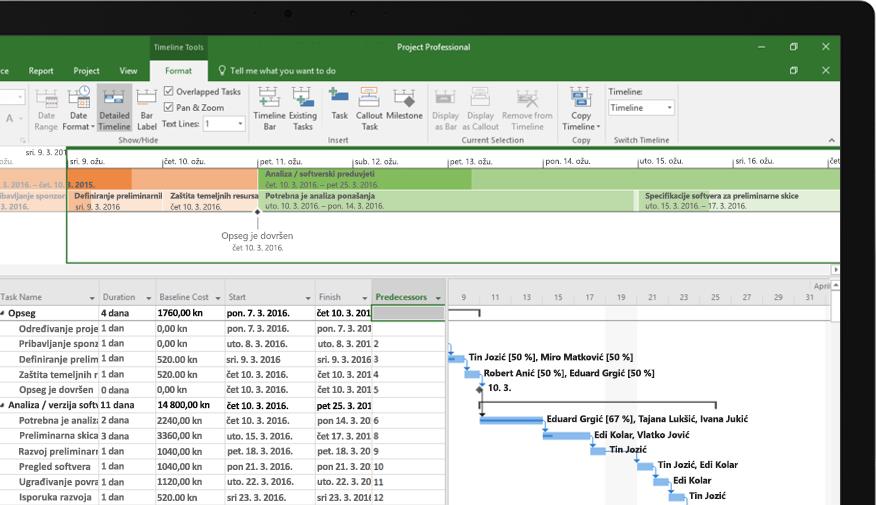
Tema diplomskog rada je izrada web aplikacije za praćenje tijeka projekta, prvenstveno namijenjena praćenju projekata studenata na završnim i diplomskim studijima. Cilj aplikacije je korisnicima olakšati postupak praćenja razvoja projekta, odnosno omogućiti da se u svakom trenutku može provjeriti u kojoj je fazi izrade sam projekt. Aplikacija će imati nekoliko korisničkih uloga, koje će ovisno o ulozi imati pristup i moći upravljati različitim dijelovima aplikacije. Sve ovo bit će izrađeno korištenjem aplikacijskog okvira ASP.NET MVC, programskog jezika C#, te CSS-a koji će se koristiti za izgled aplikacije.

U prvom poglavlju ćemo se upoznati sa još nekim od alata za praćenje tijeka razvoja projekata. Spomenuti ćemo trenutno najpoznatije alate koji se koriste u projektnom menadžmentu. Nakon toga slijedi poglavlje u kojem će se pobliže opisati korištena tehnologija, uvesti u cjelokupnu strukturu i način rada korištenjem MVC obrasca, objasniti koje prednosti pruža razvoj aplikacije u kojem su dijelovi aplikacije odvojeni. Zadnje poglavlje predstavlja programsko rješenje aplikacije, tj. sam proces izrade aplikacije od stvaranja modela, preko korištenja mehanizma Scaffolding i Code-first pristupa do dodavanja upravitelja i prikaza. U spomenutom poglavlju bit će dani i primjerci koda za određene dijelove aplikacije, te na samom kraju biti će i priložen konačan izgled aplikacije.

# SLIČNI ALATI ZA PRAĆENJE TIJEKA PROJEKTA

## Microsoft Project

**Microsoft Project** je [softverski](https://en.wikipedia.org/wiki/Project_management_software) (engl. *software*) proizvod za [upravljanje projektima](https://en.wikipedia.org/wiki/Project_management_software), koji je razvila tvrtka [Microsoft](https://en.wikipedia.org/wiki/Microsoft). Osmišljen je kako bi pomogao [voditeljima projekta](https://en.wikipedia.org/wiki/Project_manager) u izradi [plana](https://en.wikipedia.org/wiki/Plan), dodjeljivanju [resursa](https://en.wikipedia.org/wiki/Resource_(project_management)) zadacima, praćenju napretka, upravljanjem [proračunom](https://en.wikipedia.org/wiki/Budget) i analizom opterećenja. Omogućuje veoma brz početak rada te vrlo jednostavnu izvedbu projekata. Ima ugrađene predloške koji olakšavaju rad jer ne zahtijevaju stvaranje projektnog plana od početka. Također, ima i pametne alate za zakazivanje kao što su Gantt-ovi dijagrami i unaprijed popunjeni padajući izbornici koji smanjuju vrijeme trajanja obuka i čine proces planiranja projekta. Postoje i gotova izvješća koja se mogu koristiti zajednički u cijeloj tvrtki tako da svi mogu biti usklađeni. U izvješćima se mogu pronaći grafikoni s vremenskim pregledom obavljenih i preostalih zadataka, te financijska izvješća. Sve ovo omogućava voditeljima projekata i timovima održavanje visoke razine produktivnosti. Microsoft Project postao je agilan, što znači da omogućava upravljanje agilnim projektima uz pomoć jednostavnih, grafičkih ploča sa zadacima koje podržavaju Scrum, Kanban. [1][2]

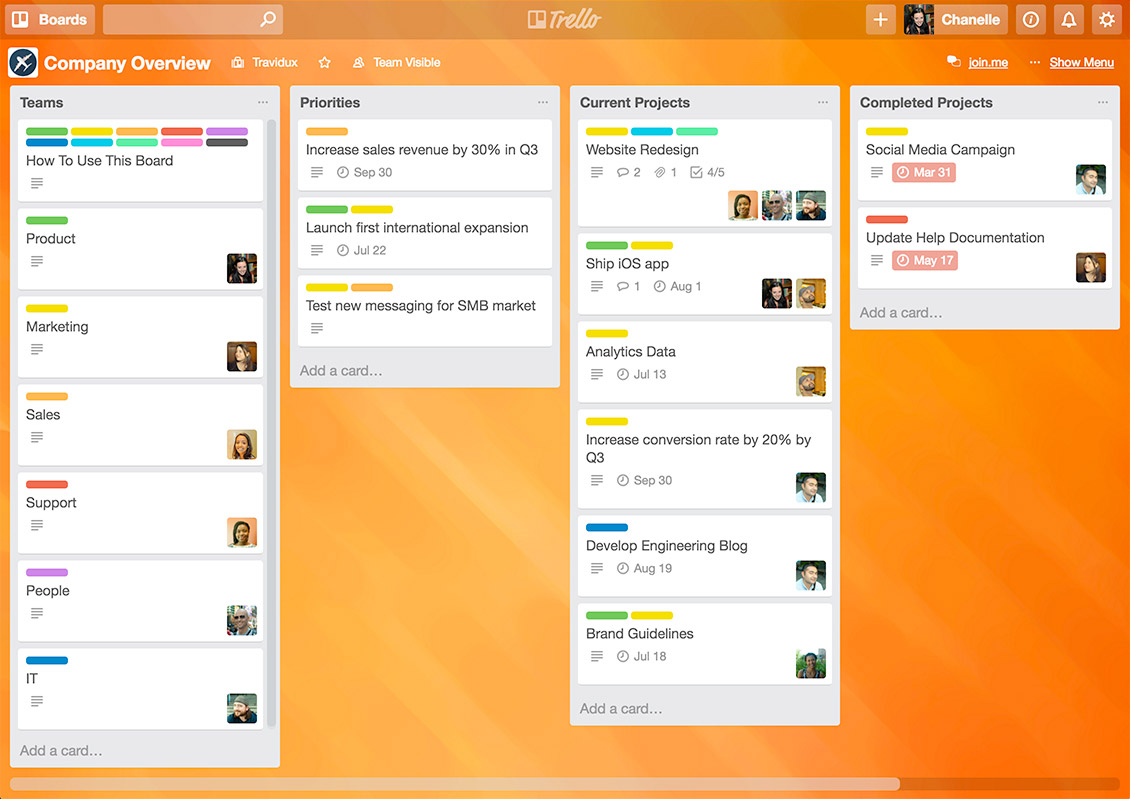


Slika 2‑1 Izgled Microsoft Project aplikacije [1]

Microsoft Project ima i svojih nedostataka. Primjerice, ukoliko korisnik želi upotrijebiti neke naprednije značajke biti će mu potreba obuka. Također, ne nudi istu razinu sofisticiranosti u modeliranju i predviđanju kao neka druga rješenja, npr. olakšava „što-ako“ analizu, ali je puno teže konfigurirati takvo nešto. To može otežati voditeljima projekata da vide „iza“ onoga što se trenutno događa i da predvide sve moguće ishode. Još jedna stvar na koju se korisnici žale je vizualizacija napretka projekta, koja po njima nije kvalitetna. Česte su pritužbe i kako su izvješća teška i dugotrajna za stvaranje. [3]

## Trello

**Trello** je alat slične namjene, služi za organizaciju i upravljanje projektima. Zamišljen je kao digitalna ploča s karticama, a koristan je kod bilo kakvih poslova u kojima je potrebno pratiti provedbu zadataka, primjerice kod razvoja softvera, posebno agilnog. Temelji se na „[Kanban](http://www.financereference.com/learn/kanban-)“[[1]](#footnote-1) metodi organizacije poslova. Naglasak je na listama i karticama koje predstavljaju kategorije i zadatke, ali nije potrebito voditi se po tom principu organizacije. To je alat koji se može prilagoditi vlastitim potrebama organiziranja. Svaka od kartica može imati sljedeće: opis, vremenski rok, dokumente, komentare, liste s oznakama dovršenosti, oznake i slično.



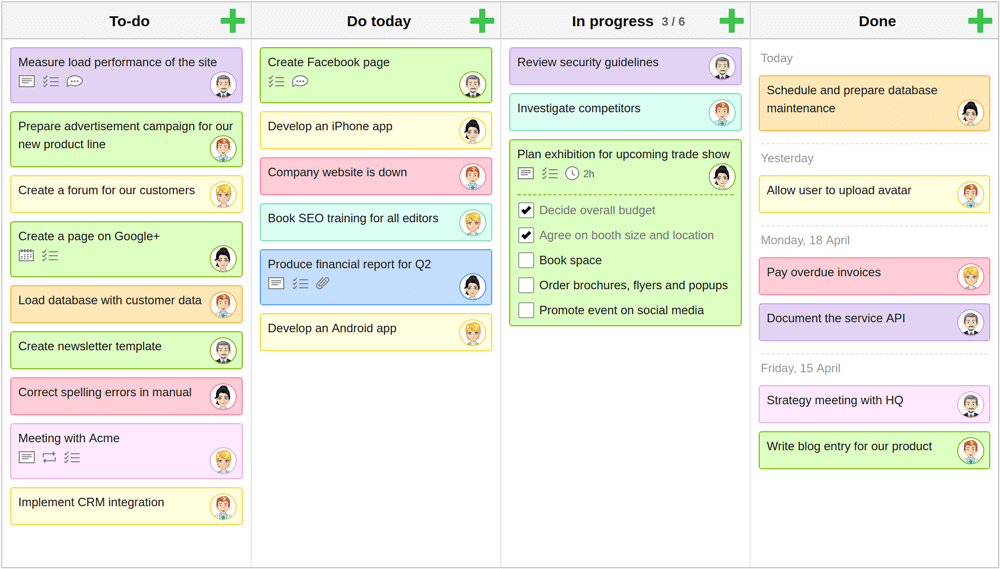
Slika 2‑2 Izgled Trello aplikacije [5]

Trello se može poistovjetiti s oglasnom pločom (npr. bijela ploča) poput one koju timovi za razvoj softvera koriste kroz [Scrum](https://en.wikipedia.org/wiki/File:Scrum_task_board.jpg)[[2]](#footnote-2) metodologiju razvoja softverskih rješenja. Trello-ve ploče, popisi i kartice omogućuju organiziranje i određivanje prioriteta projekata na zabavan, fleksibilan način. Svaka kartica se pomiče preko Trello ploče, na kojoj se nalaze liste, najčešće su to *Za napraviti, Trenutno radim, Završeno* (engl. *To Do, In Progress, Completed)*. Koristeći kontrolne liste (engl. *checklists*) i datume (engl. *due* *dates*) neće se dogoditi da se propuste važni zadaci. Trello ima sve što je potrebno za organizaciju projekata bilo koje veličine. Radi u stvarnom vremenu (engl. *real*-*time*) i to jako brzo. Kada netko pomakne karticu na ploči, svi koji koriste tu ploču odmah vide promjenu. Koristeći Trello, može se upravljati doslovce svakom vrstom projekta (pa čak i trivijalnostima kao što je kupovina u trgovini), a prilično je lako upravljati timovima, dodavati datoteke, radne liste, rokove i ostalo. Da bi stvar bila još bolja, Trello je u potpunosti besplatan, te su dostupne aplikacije za iOS i Android. [5]

Posljednji alat koji ćemo spomenuti, a koji je također našao svoju primjenu u projektnom menadžmentu je Kanban Flow.

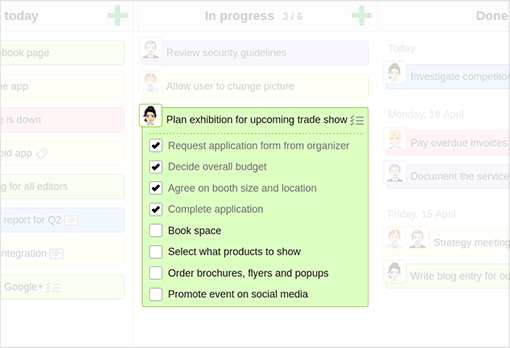
## Kanban Flow

**Kanban Flow** je aplikacija slična Trello-u, predstavlja ploču koja je podijeljena u stupce. Kratkim pogledom na ploču, dobije se pregled trenutne situacije u razvoju projekta. Stupci se mogu prilagoditi na način da se ograniči broj zadataka u svakom stupcu, te tako „prisili“ korisnika na završavanje zadataka umjesto započinjanja novih. Smanjenjem obavljanja više zadataka istovremeno (engl. *multitasking*) povećava se usredotočenost i produktivnost.



Slika 2‑3 Izgled ploče u Kanban Flow aplikaciji [7]

Svaki zadatak može se podijeliti na podzadatke te na taj način lakše pratiti napredak koji je napravljen na njemu. Kada je podzadatak završen, jednostavno se provjeri izravno na ploči. Vizualizacija rada u timskom okruženju pojednostavljuje komunikaciju i dovodi do poboljšane produktivnosti.



Slika 2‑4 Podjela zadataka na podzadatke u Kanban Flow aplikaciji [7]

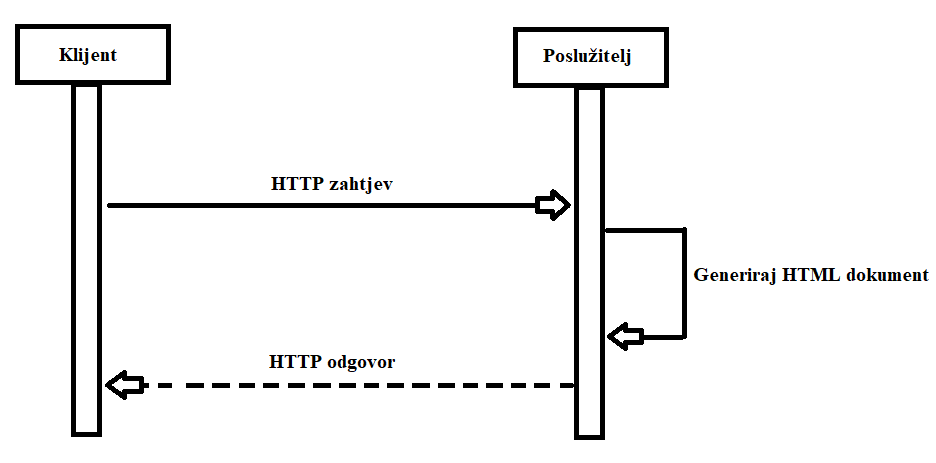
Svaka ploča može se podijeliti na timove, proizvode, poslovne prostore ili nešto drugo što će pomoći u lakšem upravljanju zadacima. Sve promjene koje se naprave na svojoj ploči odmah su dostupne svim članovima tima. Moguće je praćenje vremena utrošenog na svaki od zadataka, koristi se tzv. Pomodoro tehnika[[3]](#footnote-3) kako bi se povećala produktivnost i potaklo da se radi s punim fokusom u intervalima od 25 minuta, s kratkim stankama. Naravno, vrijeme se može upisati i ručno ili koristiti štoperica. Može se pregledati koliko je vremena potrošeno na različite zadatke, pratiti svoje vrijeme ili pogledati što rade članovi tima. Ako se žele pogledati samo zadaci koji su dodijeljeni određenoj osobi u timu ili primjerice pogledati samo zadatke označene plavom bojom, sve je to moguće koristeći filter ploče. Također, moguće je i prenositi zadatke na/iz drugih sustava pomoću značajki uvoza i izvoza. Zadaci se mogu izvesti (engl. *export* ) u Excel, XML, JSON, CSV, a uvesti (engl. *import* ) iz Excel-a i CSV-a. Ovaj alat je također besplatan i dostupne su i aplikacije za iOS i Android. [7]

# METODE I TEHNOLOGIJE KORIŠTENE U IZRADI APLIKACIJE

Za potrebu izrade ove aplikacije koristit ću ASP.NET MVC tehnologiju te Visual Studio 2017 kao integrirano razvojno okruženje tvrtke Microsoft. Cilj je prikazati i opisati mogućnosti ASP.NET MVC tehnologije, kao i drugih korištenih alata i programskih okvira na primjeru izrade ove aplikacije. Za izradu korisničkog sučelja korišteni su HTML i CSS. Za kreiranje dinamičkih web stranica sa C#-om korišten je Razor pogonski mehanizam za prikaze (engl. *view engine*). Aplikacija predstavlja mogući pristup rješenju za vođenje evidencije, odnosno praćenje tijeka projekta završnog ili diplomskog rada za studente Fakulteta elektrotehnike, strojarstva i brodogradnje u Splitu.

## ASP.NET

ASP.NET je razvojna platforma otvorenog koda (engl. *open source*)[[4]](#footnote-4) sa poslužiteljske strane (engl. *server-side*) namijenjena razvoju dinamičkih web stranica. Razvio ga je Microsoft, te je prva verzija .NET okvira (engl. *framework*) objavljena 2002. godine. To je bio nasljednik Active Server Pages (ASP) tehnologije. Na slici ispod možemo vidjeti na koji način funkcionira dohvat dinamičke web stranice, gdje poslužitelj nakon što zaprimi HTTP zahtjev, umjesto vraćanja statičkog HTML dokumenta, dinamički generira traženi dokument i vraća ga u obliku HTTP odgovora.



Slika 3‑1 Način dohvaćanja dinamičke web stranice

ASP.NET podržava niz programskih modela za izgradnju web aplikacija:

* ASP.NET Web Forme - okvir za izgradnju modularnih stranica iz komponenti, s događajima korisničkog sučelja (engl. *UI events*) koji se obrađuju na strani poslužitelja,
* ASP.NET MVC - omogućava izradu web stranica koristeći Model-View-Controller uzorak dizajna,
* ASP.NET web stranice - sintaksa za dodavanje dinamičkog koda i pristup podataka izravno unutar HTML oznaka,
* [ASP.NET Web API](https://en.wikipedia.org/wiki/ASP.NET_Web_API) - okvir za izgradnju [web API-ja](https://en.wikipedia.org/wiki/Web_API)[[5]](#footnote-5) unutar [.NET okvira](https://en.wikipedia.org/wiki/.NET_Framework)
* ASP.NET WebHooks -implementira Webhook[[6]](#footnote-6)uzorak za objavljivanje/pretplaćivanje događaja putem HTTP-a,
* SignalR - komunikacijski okvir u stvarnom vremenu (engl. *real time*) za dvosmjerno komuniciranje između klijenta i poslužitelja. [9]

## Obrazac oblikovanja model-view-controller

Model - prikaz - upravitelj (engl. *Model-View-Controller - MVC*) je obrazac oblikovanja programskih rješenja koji se u računalnoj znanosti pojavljuje već dugi niz godina. Široko je prihvaćen za razvoj aplikacija. Prednost korištenja obrasca oblikovanja MVC je odvajanje pojedinih dijelova aplikacije u zasebne komponente ovisno o njihovoj namjeni (engl. *separation of concerns within an application*), tj. odvajanje koda za vizualizaciju aplikacije (grafički prikaz), od koda za poslovnu logiku, od samih podataka i koda za korisničke akcije. Na ovaj način moguće je efikasno ponovno korištenje koda i paralelni razvoj aplikacije. MVC se sastoji od 3 zasebne cjeline: modela, prikaza i upravitelja.

Način na koji ovaj uzorak radi je sljedeći:

* HTTP zahtjevi se prosljeđuju upraviteljima koji ih obrađuju i izvode CRUD operacije nad podacima u modelu
* Podaci se kombiniraju s prikazom i vraćaju u obliku HTTP odgovora
* Na strani servera postoji model, a uključivanjem Angular[[7]](#footnote-7) okružja i sličnih tehnologija u ASP.NET MVC moguće je dio modela prebaciti i na klijenta [12]



Slika 3‑2 Način rada MVC-a [13]

### Upravitelj

Upravitelj upravlja programom i odgovoran je za njegov tok. Predstavlja prvi sloj koji se poziva kada preglednik (engl. *browser*) pozove URL. Upravlja zahtjevima korisnika (HTTP GET i HTTP POST) kada klikne na neki GUI element. Upravitelj prihvaća ulazne naputke i pretvara ih u naloge modelu ili prikazu. Upravitelji predstavljaju klase koje su pisane C# programskim jezikom. Svaka javna metoda u klasi upravitelja je akcijska metoda i ona je povezana URL-om. Kada se pošalje zahtjev URL-om koji je povezan sa akcijskom metodom, izjave u toj akcijskoj metodi se izvršavaju u modelu i prikazuje se prikaz korisniku.

### Model

Model sadrži  programske podatke kao što su informacije od objekata iz baze podataka. Sadrži stvarni sadržaj, najčešće spremljen u bazu podataka. Model se može kreirati koristeći obične C# objekte, te se njihova dosljednost može implementirati koristeći bilo koju bazu podataka, ORM okvir ili neki drugi oblik podatkovnih alata koji podržava .NET. Svi podaci se dobivaju od modela, ali se on ne može direktno pozvati, već je upravitelj taj koji od modela zahtjeva određene podatke, model obrađuje zahtjeve i vraća podatke upravitelju.

### Prikaz

Prikaz (engl. *view*) je posljednji sloj MVC arhitekture koji sadrži korisničko okruženje aplikacije. Prikazi su HTML dokumenti koji sadrže skripte, a pisani su u C# s **Razor** mehanizmom. Razor je markup jezik sa poslužiteljske strane (engl. *server side*). U razor sintaksi imamo klijentski sadržaj (engl. *client content*) i poslužiteljski sadržaj (engl. *server code*). U klijentski sadržaj ulaze HTML markup elementi, CSS kod, JavaScript i obični tekst. Razor sintaksa dopušta dodavanje poslužiteljskog koda klijentskom sadržaju, a ako postoji poslužiteljski kod na stranici, poslužitelj prvo izvršava njega i tek onda šalje stranicu pregledniku. Stranice sa Razor sintaksom imaju nastavak .cshtml, pa na taj način poslužitelj prepoznaje da se na web stranici nalazi Razor sintaksa te prvo izvršava Razor naredbe i nakon toga šalje stranicu pregledniku. [15]



Slika 3‑3 Dohvaćanje stranice u MVC arhitekturi

### Prednosti i nedostaci

Glavne prednosti MVC arhitekture već su spomenute:

* ponovna upotreba koda i podjela sustava na međusobno nezavisne cjeline, a odatle proizlazi i lakše održavanje aplikacije koja je podijeljena na više samostalnih komponenti, umjesto da je dio jednog velikog modula,
* modifikacija ne utječe na cijeli model iz razloga što modeli ne ovise o prikazima, pa se tako bilo kakve promjene u modelu neće odraziti na cijelu arhitekturu,
* kod koji je pisan na ovakav način je mnogo lakše testirati, aplikacija se može razvijati paralelno iz razloga što jedan programer može raditi na prikazima dok drugi radi na upraviteljima zaduženima za poslovnu logiku, te se na taj način povećava produktivnost jer se razvoj aplikacije može ubrzati za čak 3 puta za razliku od ostalih obrazaca dizajna web aplikacija,
* svaki model može imati više prikaza, a duplikacija koda je limitirana jer su podaci i poslovna logika u potpunosti razdvojeni.

MVC ima i neke mane:

* previše je kompleksan za implementaciju kod razvoja manjih aplikacija i njegovo korištenje u tim slučajevima dovodi do pogoršanja kako dizajna tako i performansi,
* ponekad se može dogoditi da uslijed čestih promjena modela prikaz bude preplavljen zahtjevima za izmjenu, a ukoliko on služi za prikazivanje sadržaja kojem je potrebno određeno vreme za renderiranje, česti zahtjevi za izmjenom mogu dovesti do kašnjenja,
* potrebno je znanje o većem broju tehnologija, iz tog razloga treba i veći broj programera u razvoju aplikacije. [16]

## Entity framework

Entity Framework (EF) je objektno-relacijski programski okvir (ORM), odnosno skup tehnologija u ADO.NET-u[[8]](#footnote-8) koji podržava razvoj softverskih aplikacija orijentiranih na podatke. Arhitekti i programeri podatkovno-orijentiranih aplikacija nailazili su na problem postizanja 2 cilja: modeliranje entiteta, veza među njima, poslovne logike te s druge strane rad s bazom podataka korištenom za spremanje i dohvat podataka. EF eliminira potrebu stvaranja većine programskog kôda za pristup podacima koje bez njegove primjene programer mora stvoriti, odnosno omogućuje razvojnim programerima rad s podacima u obliku objekata isvojstava specifičnih za određenu domenu, bez potrebe da se brinu za tablice baze podataka i stupce u kojima se ti podaci pohranjuju. U MVC-u svaki model predstavlja klasu, a podaci su zapisani u obliku relacijskih baza podataka, XML formatiranih datoteka i slično. Da bismo olakšali pristup tim podacima, koristi se ORM koji automatizira pristup podacima i generira kod potreban za rad s fizički pohranjenim podacima. Mapiranje podataka u objekte je implementirano u različitim ORM okruženjima. O okviru (engl. *framework*) ovisi kako se mapiraju podaci u objekte, da li se jedna relacijska tablica mapira u jednu klasu ili više, kako se mapiraju metapodaci i slično. Od ASP.net MVC verzije 4 u predlošku projekata je automatski uključen Entity Framework, a on je Microsoftov ORM. Verzija EF 6.1 je podržana u svim .NET 4+ projektima i prema svim Microsoft SQL bazama. [18]

## LINQ upiti

Bitno je spomenuti i LINQ koji predstavlja skup dodataka za .NET i olakšava rad s podacima. Definira sučelje za jednostavan pristup podacima. Upiti se obično izražavaju posebnim jezikom upita. Kod relacijskih baza podataka obično se koristi SQL (engl. Structured Query Language). U sklopu Entity Frameworka koristi se programski jezik integriranih upita - LINQ (engl. Language-Integrated Query). [19]

LINQ upiti se mogu pisati koristeći:

* metodnu sintaksu (engl. *method* *syntax*),
* sintaksu izraza upita (engl. *query* *syntax*).

Metodna sintaksa koristi se metodama za pozivanje LINQ operatora. Većina operatora prima lambda izraz kao argument koji olakšava razumijevanje i formiranje upita. Ulazni argument lambda izraza odgovara jednom ulaznom elementu kolekcije nad kojom se vrši upit. Metodna sintaksa je fleksibilna i omogućuje ulančavanje više operatora kako bi se formirali složeniji upiti. 

Kod 3‑1 Primjer metodne sintakse LINQ upita

Za razliku od metodne sintakse, sintaksa izraza upita deklarativno poziva LINQ operatore i nalikuje SQL sintaksi. Ovi upiti uvijek počinju izrazom **from**, a završavaju izrazom **select** ili **group**.



Kod 3‑2 Primjer sintakse izraza LINQ upita

Oba načina pisanja LINQ upita mogu međusobno kombinirati. Prednosti korištenja LINQ izraza u odnosu na tradicionalne SQL upite su:

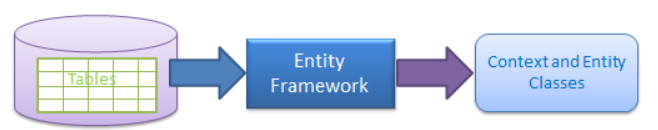
* smanjuje količinu koda
* pruža bolje razumijevanje napisanog koda
* omogućuje kreiranje upita nad podacima iz različitih izvora

## Database-first, Code-first, Model-first

Postoje 3 različita pristupa koja je moguće koristiti pri razvoju aplikacija koristeći EF:

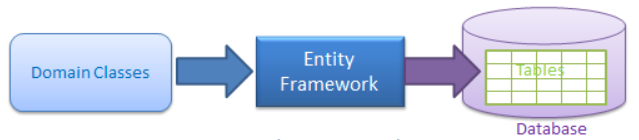
* Baza podataka-prva (engl. *database-first),*
* Kod-prvi (engl. *code-first*),
* Model-prvi (engl. *model-first*).

**Database-First** pristup je kad već postoji baza podataka ili se ona oblikuje u početnom koraku, te se tek nakon toga stvaraju ostali dijelovi aplikacije, generira se kontekst (engl. *context*) i entiteti za postojeću bazu pomoću EDM (engl. *Entity data model*) čarobnjaka koji je integriran u Visual Studio ili izvršavajući EF komande.



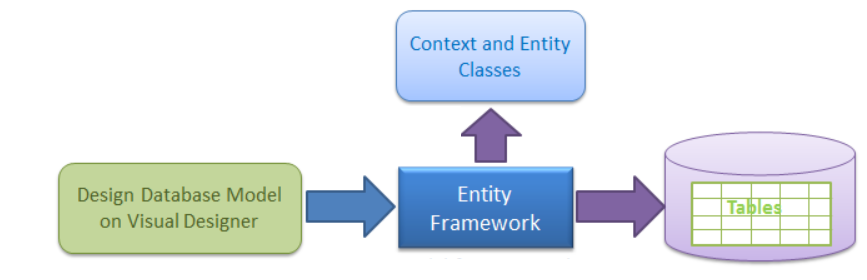
Slika 3‑4 Database first pristup [20]

**Code-First** pristup je kada nemamo postojeću bazu za aplikaciju. Najprije se stvaraju entiteti, odnosno piše se kod za klase domene i kontekstnu klasu i tek nakon toga stvara se baza podataka iz tih klasa pomoću naredbi za migraciju.



Slika 3‑5 Code first pristup [20]

**Model-First** pristup je pristup kod kojega se prvo stvaraju entiteti, odnosi među njima i nasljedna hijerarhija u vizualnom dizajneru integriranom u Visual Studio te se nakon toga generiraju entiteti, kontekstna klasa i skripta baze podataka iz vizualnog modela. [21]



Slika 3‑6 Model first pristup [20]

## Struktura direktorija aplikacije

Prilikom stvaranja ASP.NET MVC aplikacije razvojna okolina Microsoft Visual Studio automatski dodaje nekoliko datoteka i direktorija u projekt. Tako se svi upravitelji nalaze u direktoriju „Controllers“, modeli u direktoriju „Models“, a prikazi u direktoriju koji nosi ime pripadajućeg upravitelja i to na način da taj direktorij predstavlja poddirektorij „View“ direktorija. Opća struktura vidljiva je na slici ispod.



Slika 3‑7 Struktura direktorija MVC aplikacije

Tablica ispod prikazuje značaj svakog od direktorija.

Tablica 3‑1 Najvažniji direktoriji MVC aplikacije

|  |  |
| --- | --- |
| **Direktorij** | **Svrha** |
| /Controllers | Sadrži datoteke klasa za upravitelje koji obrađuju zahtjev korisnika i vraćaju odgovor. MVC zahtijeva da imena svih klasa upravitelja završavaju s “Controller” |
| /Models | Sadrži datoteke klasa modela. Obično uključuje javna svojstva koje će koristiti aplikacija za manipulaciju podacima |
| /Views | Sadrži HTML datoteke. Obično su prikazi .cshtml datoteke gdje se piše HTML i C# kod |
| /Scripts | Sadrži JavaScript datoteke. MVC po defaultu uključuje javascript datoteke za Bootstrap, jQuery[[9]](#footnote-9) i modernizr[[10]](#footnote-10) datoteku |
| /fonts | Sadrži custom fontove za našu aplikaciju |
| /Content | Sadrži statičke datoteke kao CSS datoteke, slike i ikone. MVC uključuje bootstrap.css, bootstrap.min.css i Site.css po defaultu |
| /App\_Data | Može sadržavati podatke kao što su LocalDB.mdf datoteke, xml i druge datoteke povezane s podacima naše aplikacije |
| /App\_Start | Sadrži klase koje se izvršavaju prilikom pokretanja aplikacije. Obično su tu BundleConfig.cs, FilterConfig.cs, RouteConfig po defaultu |
| /Global.asax | Omogućuje pisanje koda koji se pokreće kao odgovor na događaje na razini aplikacije, kao što su, application\_start, application\_error. |
| /packages.config | Za upravljanje paketima i verzijama koje su instalirane u aplikaciji koristeći NuGet[[11]](#footnote-11) |
| /Web.config | Sadrži konfiguracije na razini aplikacije |

Prilikom kreiranja novog ASP.NET Web Application (.NET framework) projekta, Visual Studio nudi nekoliko različitih predložaka: prazan predložak (engl. *Empty*), Web Forme (engl. *Web Forms*), MVC, Web API, Single Page Application, Azure API App predložak. Prazan predložak sadržava nekoliko osnovnih datoteka za razvoj ASP.NET Web aplikacije, ali ne dolazi sa bibliotekama i konfiguracijom potrebnom za MVC aplikaciju. Bilo koja funkcionalnost može biti naknadno dodana u prazan projekt, jedina razlika između predložaka je početni skup biblioteka, konfiguracijskih datoteka, koda i sadržaja koje Visual Studio dodaje kada kreira novi projekt.

Koncept konvencija nad konfiguracijom (engl. *convention over configuration)* je softverska dizajnerska paradigma kojom se smanjuje opseg odluka koje programeri donose tijekom razvoja aplikacije, te se na taj način povećava jednostavnost razvoja, bez smanjenja fleksibilnosti razvoja. Iz već kreiranog projekta odmah vidimo na djelu ovaj koncept kroz 3 osnovna direktorija: Controllers, Models, Views. Ova imena nije potrebno definirati u konfiguracijskoj datoteci Web.config, ona su podrazumijevana konvencijom ASP.NET MVC-a. Osnovne konvencije strukture aplikacije su:

* Upraviteljske klase imaju imena koja završavaju sa Controller, npr. HomeController i nalaze se u direktoriju Controllers,
* Postoji samo jedan Views direktorij za sve prikaze aplikacije,
* Prikaze treba spremiti u poddirektorij /Views direktorija i imenuju se po konvenciji prema imenu upravitelja bez sufiksa Controller, npr. prikazi za upravitelj AdminController će se nalaziti u direktoriju koji se zove /Views/Admin,
* Svi dijelovi sučelja koji su ponovno upotrebljivi (engl. *reusable*), tj. koriste se na više različitih mjesta u prikazima se smještaju u poddirektorij Shared Views direktorija, odnosno /Views/Shared. [25]

## Izrada korisničkog sučelja

Korisničko sučelje podijeljeno je na dva dijela. Prvi dio je predložak koji je zajednički i koristi se na svim stranicama, a u taj predložak ulaze navigacijska traka koja sadržava izbornike te podnožje (engl. *footer*) stranice. Drugi dio je sam sadržaj stranice koji se dinamički stvara za svaku stranicu. Ti predlošci nalaze se u direktoriju naziva „Views/Shared“. Unutar tog direktorija obično se nalazi *\_LoginPartial.cshtml* i \_*Layout.cshtml* datoteke koje se koriste unutar cijele aplikacije. Izgled izbornika korisničkog sučelja ovisi o vrsti korisnika koji je prijavljen. Tako na primjer postoje različiti izbornici za neprijavljenog korisnika, prijavljenog korisnika i korisnika s administratorskim ovlastima.

## ASP.NET URL rutiranje

Za pristup točno određenom resursu na mreži koriste se reference (adrese) koje jednoznačno određuju traženi resurs, to su tzv. URL adrese. Promet u ASP.NET MVC okviru započinje URL zahtjevom (engl. *requests*) te je tzv. okvir za usmjeravanje (engl. *routing framework*) jezgra svakog ASP.NET MVC zahtjeva. ASP.NET usmjeravanje (engl. *routing*) je mapiranje prema obrascu. Prilikom pokretanja, aplikacija registrira jedan ili više obrazaca usmjeravanja u tablici usmjeravanja i na taj način sustav usmjeravanja zna što učiniti sa zahtjevima koji odgovaraju tim obrascima. Rutiranje omogućava programeru definiranje URL obrasca koji se mapira u rukovatelja zahtjeva (engl. *request* *handler*). Rukovatelj zahtjevau ASP.NET-u može biti datoteka ili klasa. Rutiranje ima dvije uloge:

* Mapira URL iz HTTP zahtjeva u akciju upravitelja,
* Kreira odlazni URL koji odgovara akciji upravitelja koji će vratiti željeni prikaz.

Svaka aplikacija mora imati barem jednu rutu kako bi definirala na koji način podržava dolazni zahtjev. Konfiguracija ruta se radi metodom koja se obično zove *RegisterRoutes* i nalazi se uklasi *RouteConfig.cs,* ali se rute trebaju i registrirati nakon pokretanja aplikacije.



Kod 3‑3 RouteConfig.cs klasa za konfiguraciju ruta

Rute trebaju sadržavati upravitelj i akciju koja će se izvršiti. Parametar URL je regularni izraz kojim se pronalazi odgovarajući rukovatelj zahtjeva na osnovu URL-a u HTTP zahtjeva. Prema ovoj ruti svi dolazni zahtjevi tipa http://server/ImeKontrolera/ImeAkcije će se prevoditi u pozive metode *ImeAkcije* u klasi koja se zove ImeKontroleraController, a ukoliko je zahtjev tipa http://server/ImeKontrolera/ImeAkcije/ImeParametra onda će se predati u pozive metode *ImeAkcije* kojoj će se predati parametar *ImeParametra.*

Unutar *Global.asax.cs* datoteke nalazi se metoda *Application\_Start()* u kojoj je događaj koji predstavlja pokretanje aplikacije pa se tamo pozivom metode *RegisterRoutes* konfigurirane rute uključuju u *RouteTable* tablicu.



Kod 3‑4 Application\_Start() metoda koja se poziva pri pokretanju aplikacije

# KREIRANJE MVC WEB APLIKACIJE U .NET OKRUŽENJU

## Zahtjevi na aplikaciju

U ovom dijelu malo će se pobliže objasniti zahtjevi na aplikaciju, odnosno koje funkcionalnosti aplikacija treba imati i na koji će se način to implementirati. Bitno je spomenuti da se nastoji izraditi aplikacija koja će zamijeniti uporabu Excel tablice koja se trenutno koristi. Aplikacija će biti praktična i jednostavna za korištenje.

Kao što je već spomenuto, potrebno je izraditi web aplikaciju kojom će korisnik moći pratiti cjelokupan tijek razvoja projekta. Treba postojati više korisnika s različitim ovlastima (student, voditelj projekta, administrator). Aplikacija će omogućiti studentu unos evidencije za projekte na kojima radi (projekt, opis rada, kategorija, trajanje, broj osoba, datum) i on će moći vidjeti isključivo svoje projekte. Voditelj projekta će imati pravo pristupa samo onim projektima na kojima je on voditelj, dok administrator ima uvid u sve, odnosno ima pristup svim projektima, dodjeljuje voditelje projekata te dodaje nove korisničke uloge (engl. *roles*) koje će imati različite ovlasti.



Slika 4‑1 Excel tablica za praćenje tijeka projekta

Priložena .xls datoteka je evidencija koja se trenutno koristi, a ispunjavaju je studenti završnih godina diplomskih i preddiplomskih studija tijekom rada na diplomskom/završnom radu. Ova evidencija ima sljedeće stupce:

* **Datum** - unosi se brojčano, u formatu dd.mm.gggg,
* **Opis rada** - unosi se kratak opis svakog sastanka(npr. sastanak s mentorom u vezi dogovora oko izrade projekta),
* **Kategorija** - tijek projekta može se svrstati u različite kategorije, kao što su konzultacije, planiranje, čitanje literature, analiza zahtjeva, projektiranje, programiranje, testiranje,
* **Trajanje** - unosi se trajanje svakog dijela izrade rada,
* **Osoba** - broj osoba koje su sudjelovale u izradi dijela projekta.

Cilj nam je ovom aplikacijom omogućiti:

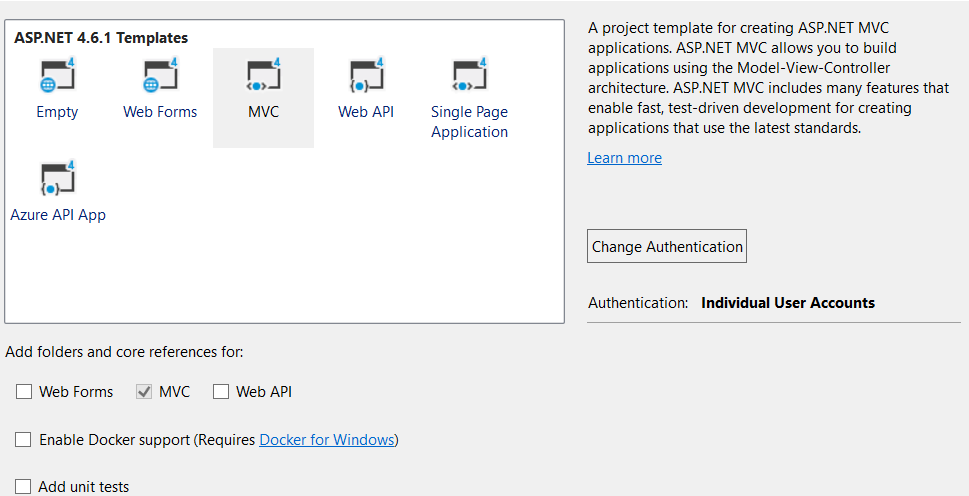
* Unos informacija o projektima, odnosno evidencije o svakoj fazi izrade projekta,
* Promjenu unesenih podataka i brisanje,
* Pregled podataka o projektima.

## ASP .NET Identity

ASP.NET Identity je sustav članstva (engl. *membership*) unutar ASP.NET okvira koji omogućava jednostavnu implementaciju autentifikacije i autorizacije korisnika. Takav sustav stvara sve potrebne tablice za korisnike i njihove uloge te stvara upravitelje, modele i prikaze koji omogućavaju korištenje registracije i prijave korisnika u aplikaciji. Osim standardnog registriranja i prijave pomoću korisničkog imena i lozinke, Identity nudi mogućnost prijave pomoću računa s društvenih mreža poput Facebooka, Twittera, i drugih. ASP.NET Identity koristi se Entity Framework okvirom i Code-First pristupom kod kreiranja potrebnih modela u bazu podataka. Korisnik na jednostavan način može promijeniti početne postavke Identity sustava, mijenjajući nazive tablica, dodavanjem ili brisanjem atributa potrebnih za logiranje i registraciju. [26]

## Vrste **korisničkih** uloga

Kako bi se osigurala sigurnost aplikacije, odnosno da bi se zaštitili podaci od neovlaštenog pristupa koristi se pristup autentifikacije i autorizacije. Ograničavanje pristupa implementirano je korištenjem korisničkih računa, te je na taj način korištenje aplikacije dopušteno isključivo korisnicima koji imaju korisnički račun. Prilikom kreiranja nove MVC aplikacije moguće je odabrati vrstu autentifikacije, ovisno o potrebama aplikacije. Ovdje će se koristiti individualni korisnički računi (engl. *Individual User Accounts*).



Slika 4‑2 Individual User Accounts autentifikacija

Za upravljanje korisničkim računima koristi se *AccountController* koji već ima postojeće metode za prijavu, odjavu i registraciju. Aplikacija koristi i prethodno definiranu klasu *IdentityModel.cs* unutar koje je definirana klasa *ApplicationUser* koja je naslijedila klasu *IdentityUser* i predstavlja entitet korisnika. Klasa je modificirana tako da je proširena potrebnim svojstvima za entitet korisnika prema potrebama aplikacije. Unutar klase dodana su 2 svojstva: *Name* i *Surname*.Ova svojstva bitna su za registraciju korisnika kada se od njega traži da prilikom registracije unese svoje ime i prezime.



Kod 4‑1 Klasa ApplicationUser

Kao što je već spomenuto postojat će 3 različite vrste korisničkih računa, odnosno 3 vrste uloga (engl. *roles*):

* Administrator – korisnik koji će imati pristup svim projektima i moći će uređivati i brisati sve podatke,
* Student – korisnik koji će unositi informacije o trenutnoj fazi svog projekta, od njega se očekuje ažuriranje svake faze projekta na kojem radi (po mogućnosti na tjednoj bazi),
* Voditelj projekta – korisnik koji će imati pristup samo onim projektima čiji je voditelj.

Za razlikovanje vrste korisnika koriste se tzv. korisničke uloge (engl. *user roles*), informacije o njima su spremljene u tablicama u bazi podataka. U tu svrhu u tablicu je ručno dodana korisnička uloga Administratora, a ostale korisničke uloge su dodane programski na način da je upravo administrator jedini koji ima pravo stvaranja novih korisničkih uloga, te kasnije dodjeljivanja korisničkih uloga ostalim korisnicima. U aplikaciji se nekim korisnicima na temelju pripadnosti korisničkoj ulozi uskraćuje pristup pojedinim dijelovima aplikacije, primjerice *Studenti* i *Voditelji* *projekata* nemaju jednaka prava kao *Administrator*. Bitno je razlikovati:

* Autentifikacija (engl. *authentication*) – preko korisničkog imena i lozinke potvrdi se identitet korisnika,
* Autorizacija (engl. *authorization*) – provjera da li korisnik ima dopuštenje za pristup određenim dijelovima aplikacije ili izvršavanje određene akcije.



Kod 4‑2 Dio koda AdminControllera-a koji dopušta pristup samo administratoru

Korištenjem atributa [*Authorize]* moguće je odobriti pristup točno određenim korisnicima, odnosno tipovima korisnika. Na gornjoj slici (Kod 4‑2) vidljivo je da pristup upravitelju *AdminController* imaju samo korisnici s administratorskim ovlastima. Isto tako slika ispod prikazuje kako je pristup metodama upravitelja *AccountController* ograničen atributom [*Authorize]*, međutim pristup metodi *SmartRegister* omogućen je atributom [*AllowAnonymus*] što znači da joj mogu pristupiti i neprijavljeni korisnici.



Kod 4‑3 Dio koda AccountController-a koji dopušta pristup prijavljenim korisnicima



Kod 4‑4 Dio koda Account Controller-a koji dopušta pristup neprijavljenim korisnicima

## Validacija podataka

Uvijek je bitna stavka provjere podataka koje korisnici unose preko elemenata korisničkog sučelja. Da bi se osigurala točnost i valjanost podataka, sav se unos treba provjeriti i onemogućiti unos krivo formatiranih podataka. Upravo u tu svrhu koriste se podatkovne anotacije (engl. *data* *annotations*). Podatkovne anotacije su atributi kojima definiramo svojstva entiteta (klasa) podatkovnog modela. Definirani su u imenskom prostoru *System.ComponentModel.DataAnnotations*. Definiraju uobičajene obrasce poput provjere raspona vrijednosti polja i provjere obaveznih polja.



Kod 4‑5 Podatkovne anotacije u modelu Project.cs



Kod 4‑6 Podatkovne anotacije korištene prilikom registracije korisnika

Tablica 4‑1 Korištene podatkovne anotacije u aplikaciji

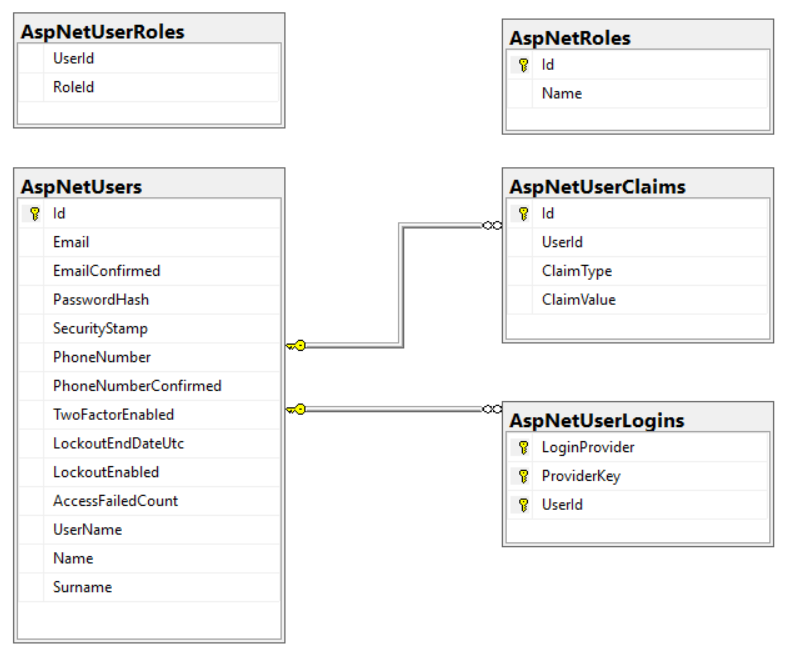
|  |  |
| --- | --- |
| **Atribut** | **Značenje** |
| [Key] | Pokazuje da je svojstvo primarni ključ |
| [Required] | Svojstvo je obavezno polje i korisnik mora unijeti valjan podatak |
| [RegularExpression] | Definira regularni izraz, odnosno koje vrijednosti smije sadržavati polje unosa za dano svojstvo |
| [DataType] | Definira koja je vrsta podataka (Date, Password, Text, Url…) |
| [Range] | Definira interval u kojem je dozvoljen unos podatka |
| [DisplayName] | Tekst koji će biti prikazan na formi |
| [StringLength] | Duljina polja tipa string |
| [Compare] | Uspoređuje vrijednosti dvaju polja (trenutnog polja i polja iz prvog parametra), te ukoliko se ne podudaraju prikazuje odgovarajuću poruku (drugi parametar) |

## Popunjavanje baze podataka korištenjem Code-First pristupa

Pri izradi web aplikacije korišten je SQL Server Object Explorer koji je verzija SQL Express baze podataka i instalirana unutar razvojne okoline Visual Studio. Već je spomenuto kako je prilikom samog stvaranja aplikacije odabrana autentifikacija putem individualnih korisničkih računa i projekt ima sljedeća 3 paketa za ASP .NET Identity:

* Microsoft.AspNet.Identity.EntityFramework
* Microsoft.AspNet.Identity.Core
* Microsoft.AspNet.Identity.Owin

Može se primijetiti da se prilikom stvaranja aplikacije sa ovakvom odabranom vrstom autentifikacije automatski dodaju 2 upravitelja: AccountController i ManageController za upravljanje korisničkim računima te 3 klase modela: AccountViewModel, IdentityModel i ManageViewModel za pristup korisničkim podacima. Odmah nakon prvog uspješno registriranog korisnika stvara se baza sa 5 tablica vidljiva na slici ispod.



Slika 4‑3 Tablice sa podacima o korisnicima i korisničkim ulogama

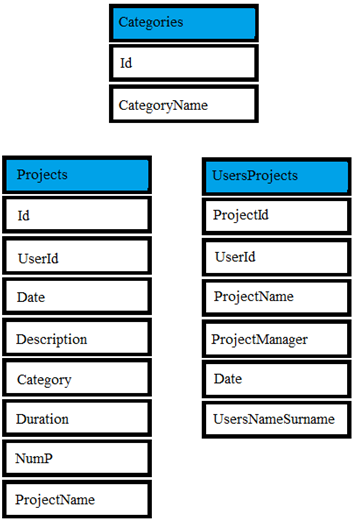
S obzirom da Entity Framework podržava stil razvoja aplikacije gdje se odmah započinje pisanjem koda bez kreiranja sheme baze u SQL-serveru, može se odmah započeti s pisanjem C# klasa, a EF iz koda zaključuje kako i gdje pohraniti instance. Za zaključivanje o shemi baze EF koriste niz konvencija:

* Za klasu objekata Project, EF pretpostavlja da te podatke pohranjuje u tablicu Projects,
* Svojstva imenovana s ID ili imeklaseID će biti mapirana u primarni ključ u tablici,
* Time je postignuto da se mapiranja ne trebaju definirati kroz metapodatke, već se kreiraju iz modela.

## Dodavanje modela

Osim već gore spomenutih tablica bilo je potrebno napraviti još neke kako bi se mogla unositi evidencija o projektima. Tako su napravljene klase za još 3 modela:

* **Projects** – tablica u kojoj se nalazi cijela evidencija za svaki projekt ,
* **UsersProjects** – tablica sa podacima o imenima projekata, njima pridjeljenim voditeljima projekata i imenima studenata koji rade na danom projektu,
* **Categories** – tablica sa svim kategorijama koje se prikazuju u formi za unos evidencije o projektu, kategorije dodaje isključivo administrator.



Slika 4‑4 Prikaz dodanih tablica i njihovih svojstava

Pogledat ćemo na primjeru modela *UsersProject* na koji način se krenulo s razvojem aplikacije. Najprije ćemo vidjeti na koji način je stvoren navedeni model s podatkovnim članovima koja predstavljaju svojstva entiteta *UsersProject*. Unutar direktorija *Models* stvorena je klasa *UsersProject* na način da se klikom na direktorij *Models* odabere opcija „Add“ i zatim „Class“ te upišemo ime modela. Nakon toga definiraju se svojstva i osigura se provjera unosa prilikom stvaranja novih objekata klase *UsersProject*.Na slici ispod prikazana je klasa i njezina svojstva.



Kod 4‑7 Model UsersProject

Pomoću atributa [Key] svojstvo *ProjectId* postavljeno je kao primarni ključ. Atributom [Required] označavamo da je svojstvo *ProjectName* obavezno, a atributom [DisplayName] je definirano kako će se u formi prikazivati tekst za dano svojstvo. [DateTime] atribut označava da će svojstvo *Date* biti u formatu datuma.

Sa Code First pristupom potrebno je definirati klasu izvedenu iz klase *System.Data.Entity.DbContext* kao „sučelje“ prema bazi. Ta klasa će sadržavati jedan ili više svojstava tipa DbSet<T> gdje će T biti tipa definirane klase entiteta s kojima želimo raditi.



Kod 4‑8 ProjectContext klasa koja predstavlja "sučelje" prema bazi

Slika iznad prikazuje *ProjectContext* klasu koja omogućava dohvaćanje i pohranu podataka iz klasa *Categories, Projects, UsersProjects*.

## View Modeli

Bitno je razlikovati *Modele* od *ViewModela*. Oboje predstavljaju klase s atributima, ali razlika je to što je *ViewModel* vrsta modela koji ima specifičnu interakciju s prikazom. On sadrži samo ona svojstva koja su potreba za prikaz. Dakle, glavni model je model entiteta koji ukoliko koristimo EF Code-first pristup je potreban kako bi se generirale tablice u bazi podataka. Stvaranjem *ViewModela* dobivamo posebnu klasu modela samo za prikaz. Samim kreiranjem aplikacije, odnosno odabirom autentifikacije generirani su neki *ViewModeli* za potrebe registracije i logiranja. Primjer takvih modela vidjet ćemo na slikama ispod u klasi *AccountViewModels* koji sadrži nekoliko *ViewModela* kao što su *LoginViewModel* i *RegisterViewModel.* [27]



Kod 4‑9 LoginViewModel

Kada se korisnik pokušava logirati u aplikaciju potrebna su samo *Email*, *Password* i *RememberMe* svojstva pa je iz tog razloga i kreiran *LoginViewModel* koji sadrži samo ta svojstva, pa nema potrebe za korištenjem modela sa dodatnim svojstvima koja se ne koriste pri logiranju.

Slično je kod *RegisterViewModela*  u kojem su definirane svojstva potrebna za registraciju korisnika, a to su: *Email, Name, Surname, Password, ConfirmPassword* što vidimo na slici ispod.



Kod 4‑10 RegisterViewModel

Još jedan primjer korištenja *ViewModela* je pri prikazivanju korisničkih uloga iz tablice *Roles.* Za tu potrebu napravljen je *RoleViewModel* sa svojstvima *Id* i *Name.*



Kod 4‑11 RoleViewModel

## Stvaranje baze podataka

Entity Framework koristi konvencije kako bi stvorio shemu baze podataka ukoliko sami ne postavimo preslikavanja iz modela u bazu podataka. Samim time, bez naznačenog povezivanja, EF će se probati povezati na *LocalDB* instancu SQL Servera i naći bazu istog imena kao klasa koja je naslijeđena iz DbContext, ovdje je to ProjectContext. Naknadno je promijenjeno ime baze u **Custom\_Identity**. Unutar *Web.config* datoteke mijenjamo postavke i to unutar *connection string-a.* ConnectionString je string koji nam daje informaciju o izvoru podataka i načinu povezivanja s njim. Najčešće se koristi kao veza s bazom podataka. To možemo vidjeti na slici ispod. [28]

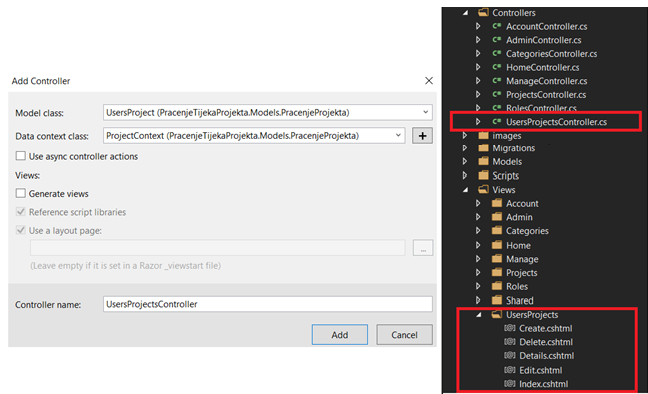


Kod 4‑12 connectionString u datoteci Web.config

## Scaffolding

**Scaffolding** je mehanizam koji je često implementiran u MVC okružjima, pa tako i u ASP.NET MVC koji omogućava automatsko generiranje osnovnog koda upravitelja i prikaza na temelju modela. Implementiran je kroz predloške koji na osnovu modela kreiraju upravitelj i povezane prikaze. Taj mehanizam zna kako će imenovati upravitelje, kako prikaze, koji kod treba dodati u koju komponentu, gdje smjestiti generirane dijelove koda u projektu aplikacije i slično. Tako automatski generirani kod predstavlja samo temelj s implementacijom najjednostavnijih mogućnosti i često nije dovoljan u izvornom obliku pa ga je potrebno modificirati i proširiti. Kako bi se dodala ova mogućnost, u sučelju razvojne okoline potrebno je odabrati direktorij *Controllers* te u kontekstnom izborniku opciju „Add“ i nakon toga jednu od ponuđenih mogućnosti. Moguće je dodati upravitelj u kojem nema automatski generiranog koda, upravitelj sa njemu pripadajućim prikazima, samo prikaz, Web API upravitelj bez automatski generiranog koda te Web API upravitelj sa automatski generiranim kodom. Na primjeru modela *UsersProject* pokazat ćemo proces stvaranja upravitelja i njemu pripadajućih prikaza. [29]

U prvom koraku odabrat ćemo „MVC 5 Controller with views, using Entity Framework“. Odabirom ove opcije omogućava da se prilikom generiranja upravitelja stvore i datoteke prikaza za osnovne akcije dodavanja, mijenjanja, brisanja podataka. U sljedećem koraku treba odabrati model prema kojemu će se generirati novi upravitelj te kontekst baze podataka i odrediti ime upravitelja.



Slika 4‑5 Odabir modela, konteksta baze podataka i imena upravitelja (lijevo), stvorena klasa upravitelja i pripadajućih prikaza (desno)

Kao rezultat, u direktoriju *Controllers* stvorena je klasa *UsersProjectsController* te je dodan direktorij *Views/UsersProjects* i prikazi za novi upravitelj što je vidljivo na Slika 4‑5.

Generirana klasa upravitelja i pripadajući prikazi sadrže sve potrebne metode za rad s bazom podataka. Nazivi generiranih datoteka i metoda su na engleskom jeziku, ali se po potrebi mogu preimenovati i unutar njih implementirati vlastita programska logika.

## Sinkronizacija modela s bazom podataka

Nakon uspješnog dodavanja modela te pripadajućeg upravitelja i prikaza, model podataka više ne odgovara shemi baze podataka. Da bi aplikacija radila ispravno, nastale promjene treba sinkronizirati u modelu podataka s postojećom bazom podataka. U tu svrhu koristi se alat *Code first migrations*. On omogućava ažuriranje postojeće baze podataka s modelom podataka nakon unesenih promjena u klasama modela. Postupak kojim se provode migracije:

1. Unutar Visual Studia otvori se sučelje konzole „Package Manager Console“.
2. Naredbom „Enable-Migrations“ omogućimo migracije unutar naše aplikacije te se automatski stvara direktorij *Migrations*
3. Naredbom „Add-Migration“ <ime\_migracije> dodaje se migracija unutar direktorija *Migrations* te se u njoj nalaze sve promjene u modelu nastale nakon zadnje migracije
4. Nakon što smo dodali migraciju potrebno je ažurirati bazu naredbom „Update-Database“ čime će se primijeniti migracije na postojeću bazu podataka.

Na ovaj način baza podataka bit će usklađena sa svim promjenama u modelima koje su nastale nakon početnog stvaranja baze. Usklađenost baze podataka i modela je nužna te se tako nakon svake promjene treba izvršiti postupak migracije. Sve datoteke koje se stvaraju prilikom izvršavanja migracija će se automatski spremati u direktorij *Migrations*.

**

Kod 4‑13 Primjer dodavanja svojstva Date u tablicu korištenjem migracija

## Način rada upravitelja

Upravitelji su središnji dio MVC predloška jer odgovaraju na zahtjeve korisnika prema aplikaciji kroz HTTP zahtjeve. Preusmjeravanje HTTP zahtjeva se radi kroz mehanizam rutiranja. Metode unutar upravitelja se zovu akcije i one se smiju biti statičke i moraju biti javne (engl. *public*). Akcije se obično povezuju sa samo jednom korisničkom aktivnošću kao klik na neki link, potvrđivanje (engl. *submit*)forme ili upisivanje URL-u u preglednik. Iz tog razloga se obično rade 2 akcije za funkcije koje se sastoje od 2 koraka. Svaka akcija predstavlja generiranje HTTP zahtjeva koji se prema URL i definiranim tablicama rutiranja povezuje s akcijom. Na primjeru akcije *Create* u upravitelju *UsersProjects* objasnit ćemo kako se ona izvodi u 2 koraka:

1. Prikaz forme
2. Prihvaćanje korisničkog ulaza

Iz tog razloga realizira se s dvije akcije:

* Jedna akcija služi za prikaz forme preko HttpGet (Kod 4‑14)
* Druga akcija prihvaća podatke preko HttpPost(Kod 4‑15)



Kod 4‑14 HttpGet Create akcija



Kod 4‑15 HttpPost Create akcija

Selektor akcijskih glagola (npr. [HttpPost]) (engl. a*ctionverbs selector*) je atribut koji se koristi za definiranje izbora akcija na osnovu metode u HTTP zahtjevu. U prethodnom primjeru vidimo dvije akcije istog imena, ali se jedna poziva kada je u HTTP zahtjevu navedena metoda GET, a druga ako je u HTTP zahtjevu POST metoda.

## Prenošenje podataka iz upravitelja u prikaz

Uobičajeni način korištenja modela u prikazu se kreira pozivom funkcije *View(model objekt)* u upravitelju.. To je tzv. *strongly-typed* prikaz kojim su čvrsto postavljeni tipovi podataka u prikazu. Drugi način je korištenje *ViewBag* objekta. To je dinamički objekt koji je na raspologanju samo promatranoj akciji. On je zamjena za *ViewData* objekt, međutim kod *ViewBag* objekta nema potrebe za tipiranjem podataka i slično. ViewBag to sve radi automatski, čak i za kompleksne podatke. Postoji i *TempData* dinamički objekt sa string ključevima i objekt vrijednostima, ali je na raspolaganju i na sljedećoj stranici kod redirekcije. Realizira se preko ograničene sesije za HTTP zahtjev. *TempData* ovdje nismo koristili, ali *ViewBag* objekt jesmo te ćemo primjer korištenja pokazati na slikama ispod.



Kod 4‑16 Dio koda iz AdminController-a u kojem se spremaju podaci u ViewBag.Roles i ViewBag.Users

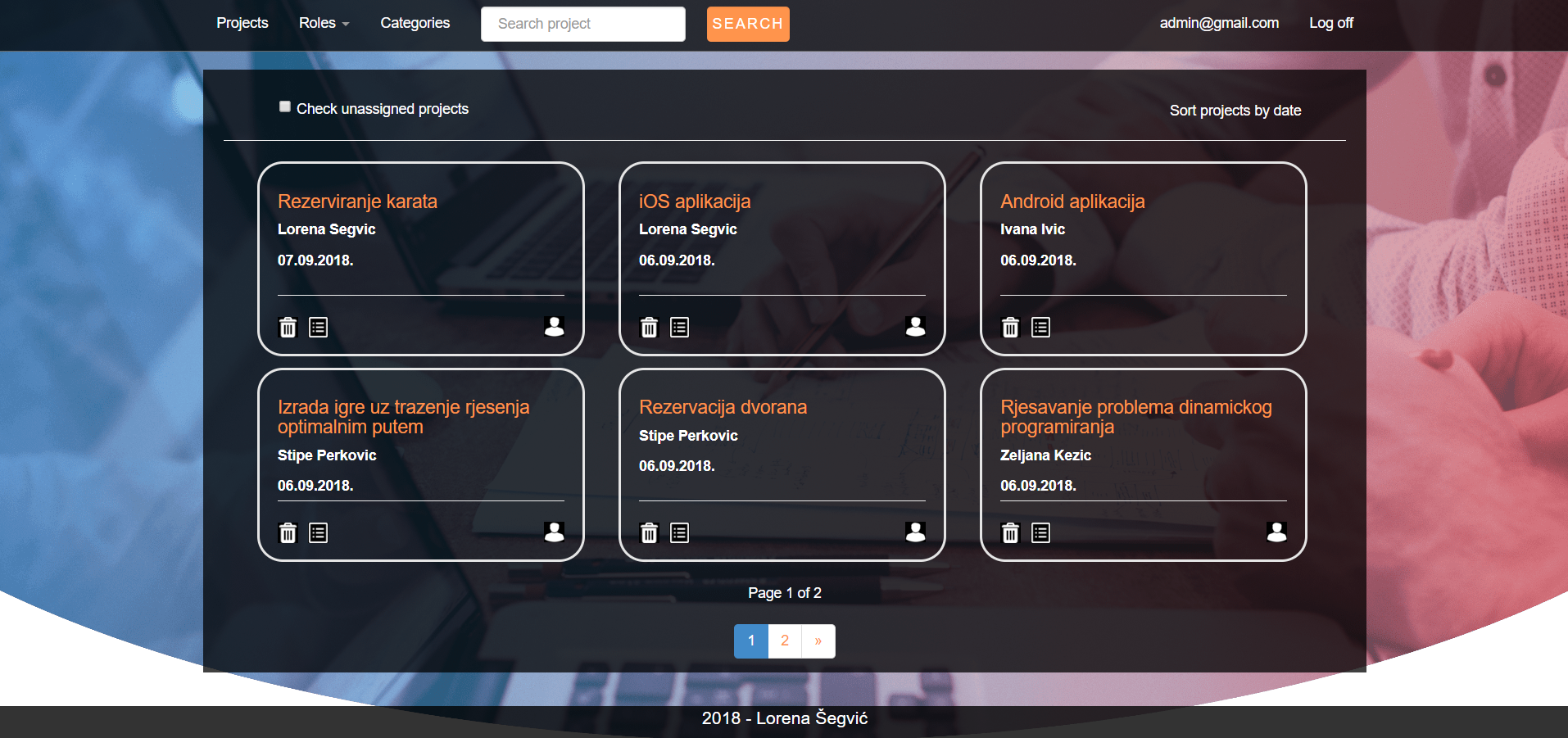


Kod 4‑17 Korištenje ViewBag-ova definiranih u upravitelju u prikazu akcije AssignRole()

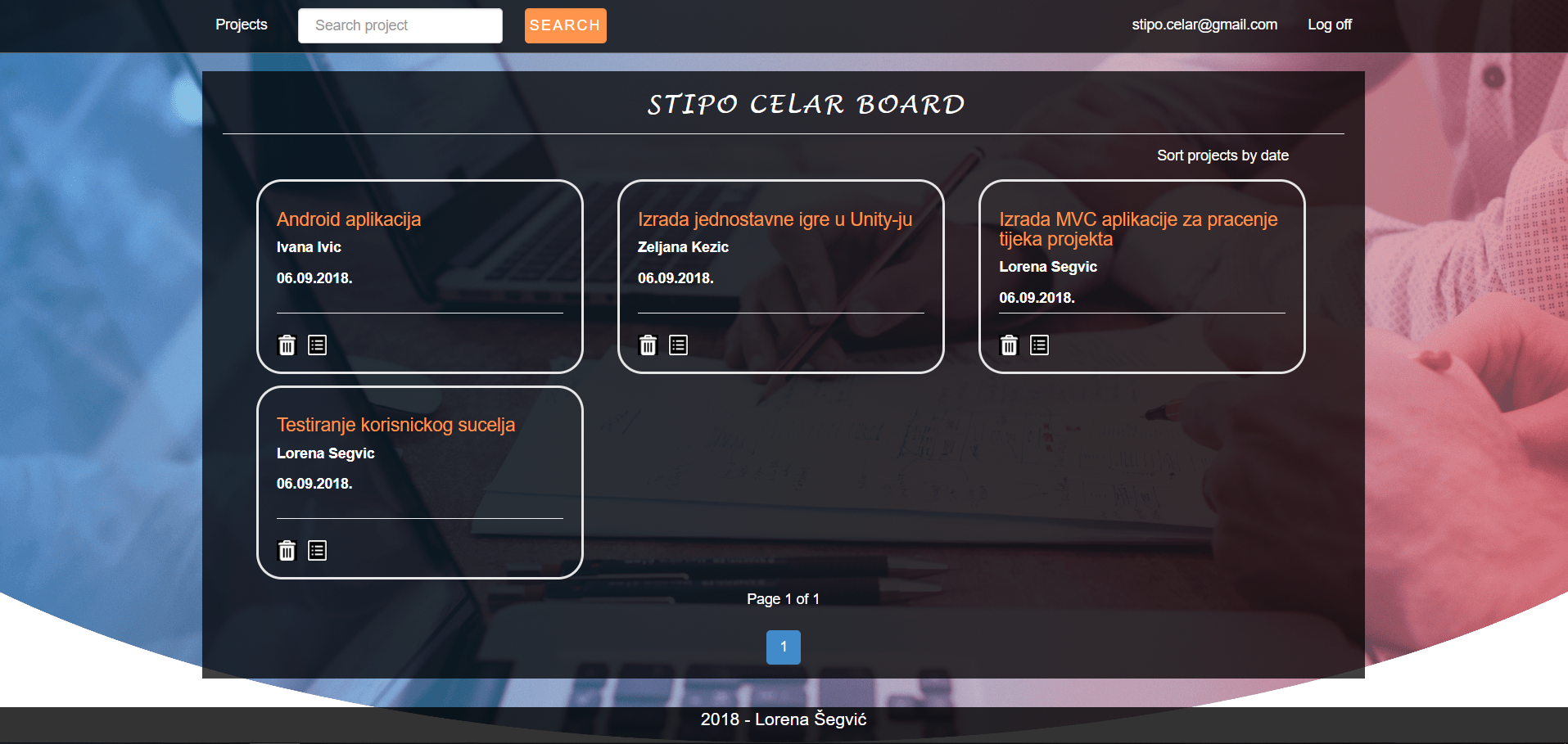
## Izgled aplikacije

Konačan izgled aplikacije možemo vidjeti na slikama ispod. Slike prikazuju neke najvažnije stranice, a to su redom:

* Prikaz za administratora – prikaz svih projekata unutar aplikacije(Slika 4‑6),
* Prikaz za voditelja projekta – vidi samo projekte koji su mu dodijeljeni, odnosno projekte na kojima je voditelj(Slika 4‑7),
* Prikaz za sve korisničke uloge – unos evidencije o tijeku projekta(Slika 4‑8),
* Prikaz za studenta – prikaz samo njegovih projekata(Slika 4‑9),
* Unos evidencije studenta na odabranom projektu(Slika 4‑10).



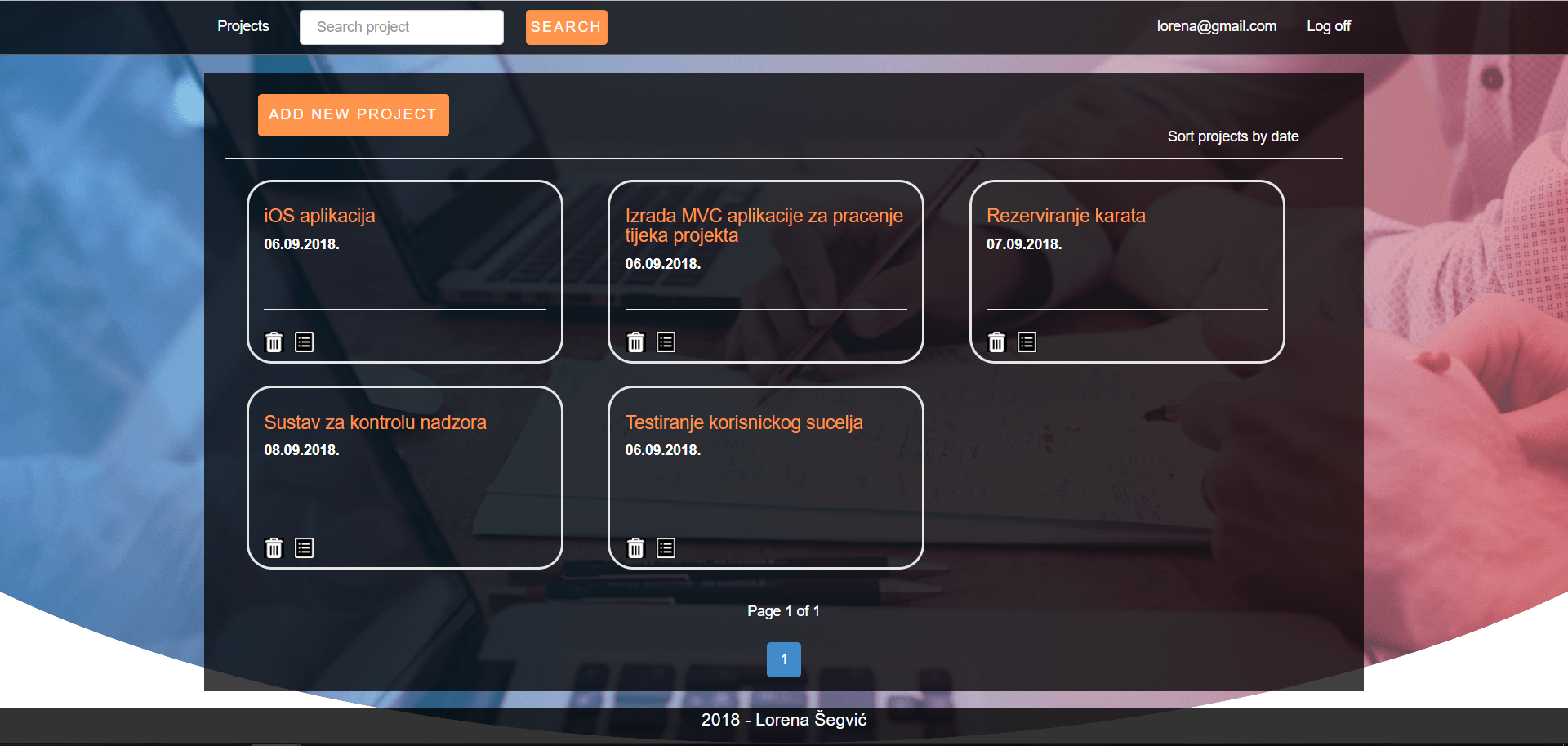
Slika 4‑6 Prikaz za administratora



Slika 4‑7 Prikaz za voditelja projekta



Slika 4‑8 Prikaz evidencije za odabrani projekt



Slika 4‑9 Prikaz projekata trenutno ulogiranog studenta



Slika 4‑10 Unos evidencije studenta za odabrani projekt

# ZAKLJUČAK

Ovaj rad je rezultirao uspješnom izradom web aplikacije za praćenje tijeka projekta koja nudi moderniji i pristupačniji način za razliku od dosada korištene excel tablice u kojoj se bilježila evidencija o razvoju projekta. Izrađena aplikacija koja je razvijena koristeći MVC predložak i C# programski jezik unutar ASP .NET tehnologije omogućuje korisnicima lakše praćenje razvoja projekata kroz intuitivno korisničko sučelje i niz funkcionalnosti koje ima. U samoj aplikaciji postoji više vrsta korisnika s različitim ovlastima, a samim time i pristupom različitim dijelovima aplikacije.

Kroz rad pokazano je kako MVC predložak omogućuje da se svaki segment web sadržaja izrađuje zasebno, a isto tako na koji način se kreira baza koristeći Entity Framework i Code-first pristup. Upravo zbog korištenja ovakvog obrasca razvoja aplikacije gdje postoji podjela na modele, upravitelje i prikaze, omogućena je višeslojnost aplikacije i modularnost programskog koda, a time se ujedno i pospješuje lakša nadogradnja aplikacije u budućnosti.

Iako ova aplikacija predstavlja samo prototip s implementiranim rješenjem za određene korisničke uloge, ona se može dodatno nadograditi tako da se dodaju i ostale uloge ukoliko se radi na većim projektima. Aplikacija nudi moguće rješenje za praćenje tijeka projekata studenata na završnim i diplomskim radovima na Fakultetu elektrotehnike, strojarstva i brodogradnje u Splitu.

# LITERATURA

1. „Project and portfolio management software“, s Interneta, https://products.office.com/hr-hr/project/project-and-portfolio-management-software, zadnji pristup: 16. kolovoza 2018.
2. „Microsoft Project“, s Interneta, https://en.wikipedia.org/wiki/Microsoft\_Project, zadnji pristup: 1. kolovoza 2018.
3. „Investing in Microsoft Project Software: The pros and cons“, s Interneta, http://prosymmetry.com/investing-in-microsoft-project-software-the-pros-and-cons/, zadnji pristup: 11. rujna 2018.
4. „Kanban metoda“, s Interneta, https://www.info-novitas.hr/o-nama/metodologije-rada/kanban-metoda/, zadnji pristup: 14. kolovoza 2018.
5. „Trello“, s Interneta, https://trello.com/, zadnji pristup: 14. kolovoza 2018.
6. „Scrum(software development)“, s Interneta, https://en.wikipedia.org/wiki/Scrum\_(software\_development), zadnji pristup: 18. srpnja 2018.
7. „Kanban Flow“, s Interneta, https://kanbanflow.com/, zadnji pristup: 14. kolovoza 2018.
8. „Open-source software“, s Interneta, https://en.wikipedia.org/wiki/Open-source\_software, zadnji pristup: 5. kolovoza 2018.
9. „ASP.NET“, s Interneta, https://en.wikipedia.org/wiki/ASP.NET, zadnji pristup: 18. kolovoza 2018.
10. „Web API“, s Interneta, https://en.wikipedia.org/wiki/Web\_API, zadnji pristup: 30. srpnja 2018.
11. „New in version 9 for developers“, s Interneta, https://docs.microsoft.com/hr-hr/dynamics365/get-started/whats-new/customer-engagement/new-in-version-9-for-developers, zadnji pristup: 20. kolovoza 2018.
12. „Model-View-Controller“, s Interneta, https://hr.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller, zadnji pristup: 21. kolovoza 2018.
13. „Introduction to ASP.NET MVC“, s Interneta, https://channel9.msdn.com/Series/Introduction-to-ASP-NET-MVC/01, zadnji pristup: 28. kolovoza 2018.
14. „Angular (application platform)“, s Interneta, https://en.wikipedia.org/wiki/Angular\_(application\_platform), zadnji pristup: 12. kolovoza 2018.
15. „MVC: Model, View, Controller“, s Interneta, https://www.codecademy.com/articles/mvc, zadnji pristup: 20. kolovoza 2018.
16. „What is MVC? Advantages and Disadvantages of MVC“, s Interneta, https://www.interserver.net/tips/kb/mvc-advantages-disadvantages-mvc/, zadnji pristup: 21. kolovoza 2018.
17. „ADO.NET“ https://en.wikipedia.org/wiki/ADO.NET, zadnji pristup: 16.8.2018.
18. „Entity Framework“, s Interneta, https://en.wikipedia.org/wiki/Entity\_Framework#Entities, zadnji pristup: 18. kolovoza 2018.
19. „Language-Integrated Query (LINQ)“ , s Interneta, https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/, zadnji pristup: 15. kolovoza 2018.
20. „Entity Framework Tutorial“, s Interneta, http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx, zadnji pristup: 29. kolovoza 2018.
21. “Development Approaches with Entity Framework“, s Interneta, http://www.entityframeworktutorial.net/choosing-development-approach-with-entity-framework.aspx, zadnji pristup: 22. kolovoza 2018.
22. „Zašto je jQuery najpopularnija JavaScript biblioteka?“, s Interneta, https://www.popwebdesign.net/popart\_blog/2014/02/sta-je-jquery/, zadnji pristup: 23. kolovoza 2018.
23. „Modernizr“, s Interneta, https://en.wikipedia.org/wiki/Modernizr, zadnji pristup: 21. kolovoza 2018.
24. „Create .NET apps faster with NuGet“, s Interneta, https://www.nuget.org/, zadnji pristup: 15. kolovoza 2018.
25. „ASP .NET MVC Folder Structure“, s Interneta, http://www.tutorialsteacher.com/mvc/mvc-folder-structure, zadnji pristup: 21. kolovoza 2018.
26. „ASP .NET Identity“, s Interneta, https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity, zadnji pristup: 27. kolovoza 2018.
27. „Difference between Model/Domain Model and ViewModel in MVC“, s Interneta, https://dotnet-helpers.com/mvc/difference-between-modeldomain-model-and-viewmodel-in-mvc/, zadnji pristup: 27.kolovoza 2018
28. „Connection string“, s Interneta, https://en.wikipedia.org/wiki/Connection\_string, zadnji pristup: 29. kolovoza 2018.
29. „ASP .NET MVC - Scaffolding“, s Interneta, https://www.tutorialspoint.com/asp.net\_mvc/asp.net\_mvc\_scaffolding.htm, zadnji pristup: 24. rujna 2018.

# SAŽETAK/ABSTRACT I KLJUČNE RIJEČI/KEYWORDS

**Sažetak**

*Kroz ovaj diplomski rad opisan je cjelokupan proces izrade web aplikacije za praćenje tijeka projekta korištenjem MVC (Model-View-Controller) obrasca oblikovanja. Prilikom samog razvoja aplikacije korištene su različite tehnologije bazirane na .NET-u te je opisana struktura aplikacije i njena realizacija u ASP .NET tehnologiji. U prvom dijelu rada upoznajemo se sa sličnim alatima čija je namjena također praćenje razvoja projekata. Nakon toga slijedi upoznavanje sa korištenom tehnologijom, a poglavito sa MVC obrascem kod kojeg se odvaja prezentacija sadržaja aplikacije od ostatka aplikacije. U zadnjem dijelu, ujedno i najvažnijem za praktični dio rada, pokazano je na koji način je tekao razvoj aplikacije koristeći Code-first pristup za izradu baze podataka, koje korisničke uloge postoje u aplikaciji i koje ovlasti imaju, te je općenito opisan način dodavanja Upravitelja i Prikaza pomoću mehanizma Scaffolding.*

**Ključne riječi**

*ASP .NET, Model, Upravitelj, Prikaz, Entity Framework, Code-first, Scaffolding*

**Web application for project tracking**

**Abstract**

*In this graduate thesis, the entire process of developing of web application for monitoring project flow was described using the MVC (Model-View-Controller) pattern. During the development of the application, various technologies based on .NET were used and the structure of the application and its implementation in ASP .NET technology were described. In the first part of the paper, we briefly mentioned similar tools whose purpose is also to monitor project development. After that, we briefly mentioned the technology used in project, especially with the MVC pattern that separates the application content presentation from the rest of the application. In the practical part of the paper we described in detail how the application is developed using the Code-first approach for database creation, which user roles exist in the application and which authorities they have, and how to add Controllers and Views using the Scaffolding mechanism.*

**Keywords**

*ASP .NET, Model, Controller, View, Entity Framework, Code-first, Scaffolding*

# PRILOZI

## Kazalo slika, tablica i kodova

### Kazalo slika

[Slika 2‑1 Izgled Microsoft Project aplikacije [1] 3](#_Toc524462288)

[Slika 2‑2 Izgled Trello aplikacije [5] 4](#_Toc524462289)

[Slika 2‑3 Izgled ploče u Kanban Flow aplikaciji [7] 6](#_Toc524462290)

[Slika 2‑4 Podjela zadataka na podzadatke u Kanban Flow aplikaciji [7] 6](#_Toc524462291)

[Slika 3‑1 Način dohvaćanja dinamičke web stranice 8](#_Toc524462292)

[Slika 3‑2 Način rada MVC-a [13] 10](#_Toc524462293)

[Slika 3‑3 Dohvaćanje stranice u MVC arhitekturi 11](#_Toc524462294)

[Slika 3‑4 Database first pristup [20] 14](#_Toc524462295)

[Slika 3‑5 Code first pristup [20] 15](#_Toc524462296)

[Slika 3‑6 Model first pristup [20] 15](#_Toc524462297)

[Slika 3‑7 Struktura direktorija MVC aplikacije 16](#_Toc524462298)

[Slika 4‑1 Excel tablica za praćenje tijeka projekta 21](#_Toc524462299)

[Slika 4‑2 Individual User Accounts autentifikacija 23](#_Toc524462300)

[Slika 4‑3 Tablice sa podacima o korisnicima i korisničkim ulogama 28](#_Toc524462301)

[Slika 4‑4 Prikaz dodanih tablica i njihovih svojstava 29](#_Toc524462302)

[Slika 4‑5 Odabir modela, konteksta baze podataka i imena upravitelja (lijevo), stvorena klasa upravitelja i pripadajućih prikaza (desno) 34](#_Toc524462303)

[Slika 4‑6 Prikaz za administratora 38](#_Toc524462304)

[Slika 4‑7 Prikaz za voditelja projekta 38](#_Toc524462305)

[Slika 4‑8 Prikaz evidencije za odabrani projekt 39](#_Toc524462306)

[Slika 4‑9 Prikaz projekata trenutno ulogiranog studenta 39](#_Toc524462307)

[Slika 4‑10 Unos evidencije studenta za odabrani projekt 40](#_Toc524462308)

### Kazalo tablica

[Tablica 3‑1 Najvažniji direktoriji MVC aplikacije 16](#_Toc524462309)

[Tablica 4‑1 Korištene podatkovne anotacije u aplikaciji 27](#_Toc524462310)

### Kazalo kodova

[Kod 3‑1 Primjer metodne sintakse LINQ upita 13](#_Toc524462311)

[Kod 3‑2 Primjer sintakse izraza LINQ upita 14](#_Toc524462312)

[Kod 3‑3 RouteConfig.cs klasa za konfiguraciju ruta 19](#_Toc524462313)

[Kod 3‑4 Application\_Start() metoda koja se poziva pri pokretanju aplikacije 20](#_Toc524462314)

[Kod 4‑1 Klasa ApplicationUser 23](#_Toc524462315)

[Kod 4‑2 Dio koda AdminControllera-a koji dopušta pristup samo administratoru 24](#_Toc524462316)

[Kod 4‑3 Dio koda AccountController-a koji dopušta pristup prijavljenim korisnicima 25](#_Toc524462317)

[Kod 4‑4 Dio koda Account Controller-a koji dopušta pristup neprijavljenim korisnicima 25](#_Toc524462318)

[Kod 4‑5 Podatkovne anotacije u modelu Project.cs 26](#_Toc524462319)

[Kod 4‑6 Podatkovne anotacije korištene prilikom registracije korisnika 27](#_Toc524462320)

[Kod 4‑7 Model UsersProject 30](#_Toc524462321)

[Kod 4‑8 ProjectContext klasa koja predstavlja "sučelje" prema bazi 30](#_Toc524462322)

[Kod 4‑9 LoginViewModel 31](#_Toc524462323)

[Kod 4‑10 RegisterViewModel 32](#_Toc524462324)

[Kod 4‑11 RoleViewModel 32](#_Toc524462325)

[Kod 4‑12 connectionString u datoteci Web.config 33](#_Toc524462326)

[Kod 4‑13 Primjer dodavanja svojstva Date u tablicu korištenjem migracija 35](#_Toc524462327)

[Kod 4‑14 HttpGet Create akcija 36](#_Toc524462328)

[Kod 4‑15 HttpPost Create akcija 36](#_Toc524462329)

[Kod 4‑16 Dio koda iz AdminController-a u kojem se spremaju podaci u ViewBag.Roles i ViewBag.Users 37](#_Toc524462330)

[Kod 4‑17 Korištenje ViewBag-ova definiranih u upravitelju u prikazu akcije AssignRole() 37](#_Toc524462331)

## Popis oznaka i kratica

XML EXtensible Markup Language

JSON JavaScript Object Notation

CSV Comma-separated values

MVC Model-View-Controller

HTML HyperText Markup Language

CSS Style Sheets

URL Uniform Resource Locator

ORM Object-Relational Mapping

GUI Graphical user interface

ADO.NET ActiveX Data Objects .NET

API Application Programming Interface

CRUD [Create, read, update, delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete)

OWIN Open Web Interface for .NET

LINQ Language Integrated Query

SQL Structured Query Language

1. Kanban je metoda namijenjena razvoju software-skih proizvoda i procesa, ali s naglaskom na just-in-time isporuku. Developeri izvlače posao iz redova i proces je do isporuke korisniku vidljiv svima koji sudjeluju u razvoju. Kod ove metode postoji posao koji čeka u redu te prolazi kroz faze dok ne bude završen. Kada je posao u fazi završen tada prelazi u drugu fazu. Ako netko treba posao jednostavno ga povuče iz prethodne faze. [4] [↑](#footnote-ref-1)
2. Scrum pristup je brzi, prilagodljivi i samoorganizirani proces razvoja programskog proizvoda. Ovaj pristup koji povećava brzinu te fleksibilnost u razvoju programskog proizvoda. Proces razvoja provodi se kroz tri faze: • Prije igre (planiranje, dizajn/arhitektura visoka razina apstrakcije) • Igra (razvoj, sprintovi – iterativni ciklusi, poboljšanja, nove verzije) • Poslije igre (nema novih zahtjeva, sustav spreman za produkciju). Razvoj se sastoji od iteracija (sprinteva) te svaka iteracija traje dva do četiri tjedna, ali to trajanje mora biti fiksno i unaprijed definirano, tijekom kojih se izrađuje verzija proizvoda spremna za isporuku dok cjelokupni projekt (jedna igra) traje tri do osam sprinteva. [6] [↑](#footnote-ref-2)
3. Cilj Pomodoro tehnike je da se radi fokusirano u kratkim vremenskim intervalima. Čuvanjem fokusa obavlja se više posla u manje vremena. Osnovni koraci su: 1. odlučivanje na kojem će se zadatku raditi, 2. raditi 25 minuta sa punim fokusom, 3. uzeti kratku pauzu (3-5 minuta), 4. ponovno raditi 25 minuta do iduće pauze i tako sve do kraja, 5. svaka četiri Pomodoro ciklusa uzeti dužu pauzu (15-30 minuta). [7] [↑](#footnote-ref-3)
4. Otvoreni softver (*engl. Open source software*) je [softver](https://sh.wikipedia.org/wiki/Softver) čiji je [izvorni kod](https://sh.wikipedia.org/wiki/Izvorni_kod) dostupan unutar "open source" licence svim korisnicima, te ga mogu mijenjati, prepravljati i poboljašvati njegov sadržaj. To znači da uz 'open source' [programe](https://sh.wikipedia.org/wiki/Kompjuterski_program) dolazi i čitav izvorni kod u nekom programskom jeziku, što nije slučaj sa plaćenim softverom [8] [↑](#footnote-ref-4)
5. Web API je aplikacijsko programsko sučelje za web poslužitelje i web preglednike. To je koncept [razvoja web stranica](https://en.wikipedia.org/wiki/Web_development) , koji se obično ograničava na klijentsku stranu [web aplikacije](https://en.wikipedia.org/wiki/Web_application) (uključujući sve [web okvire koji](https://en.wikipedia.org/wiki/Web_framework) se koriste), te tako obično ne uključuje detalje implementacije web poslužitelja ili preglednika kao što su [SAPI](https://en.wikipedia.org/wiki/Server_Application_Programming_Interface) ili API, osim ako nije javno dostupan. [10] [↑](#footnote-ref-5)
6. WebHooks je HTTP uzorak za povezivanje Web API-ja i usluga s modelom za objavljivanje/pretplaćivanje. Pošiljatelji s rješenja WebHooks obavještavaju primatelje o događajima slanjem zahtjeva krajnjim točkama primatelja s određenim informacijama o događajima. [11] [↑](#footnote-ref-6)
7. AngularJS je JavaScript MVC framework napravljen od strane Google-a koji omogućava izradu dobro strukturiranih, lakih za testiranje i održavanje front-end aplikacija. Dozvoljava korištenje HMTL-a i CSS-a, ali isto tako proširuje mogućnosti HTML-a. Angular komponente su jako korisne i relativno lako ih je koristiti. [14] [↑](#footnote-ref-7)
8. ADO.NET (ActiveX Data Objects .NET) je skup komponenti računalnih softvera koje programeri mogu koristiti za pristup podacima i podatkovnim uslugama iz baze podataka. To je dio [biblioteke osnovnog razreda](https://en.wikipedia.org/wiki/Base_Class_Library) koji je uključen u Microsoft .NET Framework. [17] [↑](#footnote-ref-8)
9. JQuery je jednostavna JavaScript biblioteka koja pojednostavljuje upotrenu JavaScript-a i omogućava rukovanje događajima (engl. *events*), kreiranje animacija, Ajax interakciju i drugo. Zamišljena je da promjeni način na koji se piše JavaScript kod. Službeni jQuery slogan glasi: "Piši manje, učini više" što znači da je jednom linijom jQuery koda moguće pozvati funkcionalnosti za koje inače treba mnogo više rada ako se koristi klasično JavaScript kodiranje [22] [↑](#footnote-ref-9)
10. Modernizr je [JavaScript](https://en.wikipedia.org/wiki/JavaScript)[biblioteka](https://en.wikipedia.org/wiki/Library_(computer_science)) koja je dizajnirana za otkrivanje [HTML5](https://en.wikipedia.org/wiki/HTML5) i [CSS3](https://en.wikipedia.org/wiki/CSS_3) značajki u različitim [preglednicima](https://en.wikipedia.org/wiki/Web_browser), što omogućava izbacivanje upotrebe nekorištenih značajki. [23] [↑](#footnote-ref-10)
11. NuGet je paket menadžer (engl. *package manager*) za .NET.  NuGet je centralno spremište repozitorija koje koriste svi autori paketa i potrošači. [24] [↑](#footnote-ref-11)